SMS Scam Detection

1. Chosen Model & Why

Models:

- **Arabic-pretrained BERT:** Captures deep linguistic nuances and context in Arabic SMS messages, essential for distinguishing legitimate from scam messages.
- **Convolutional Neural Networks (CNNs):** Detect structural patterns in the message content by extracting local features.

Why this combination:

- BERT provides strong contextual understanding of Arabic text.
- CNNs complement BERT by detecting local features and structural patterns that may indicate scams.

Detailed Steps:

1. Data Collection (see Dataset section below)
2. Preprocessing (see Dataset section below)
3. Model Development:
    - Train BERT on tokenized SMS messages.
    - Use CNN layers on top of embeddings to detect local patterns.
4. Optimization (see section 3)

2. Dataset

What is the Dataset?

- Contains SMS messages labeled as spam or ham (legitimate).
- Sources:
    1. Mendeley SMS Spam Collection
    2. Arabic SMS Dataset on GitHub
    3. UCI SMS Spam Collection

Why chosen:

- Real-world examples of SMS scams.
- Balanced variety of spam and ham messages.

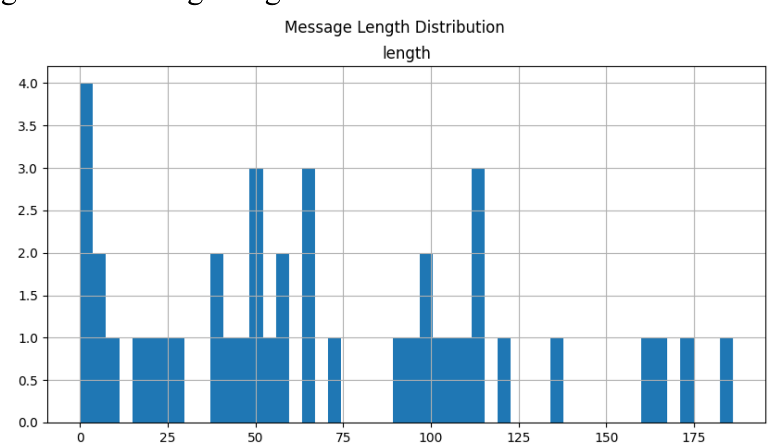
- Covers Arabic-specific challenges like diacritics, morphology, and abbreviations.
- Widely used in academic research for benchmarking.

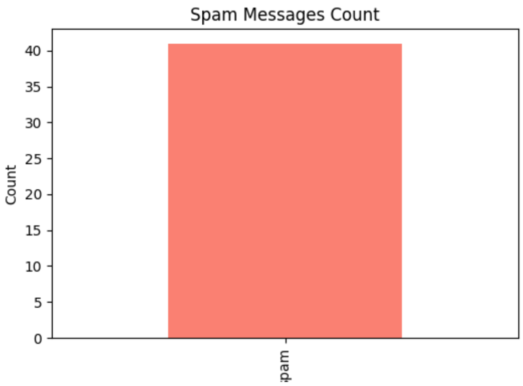
Preprocessing Steps:

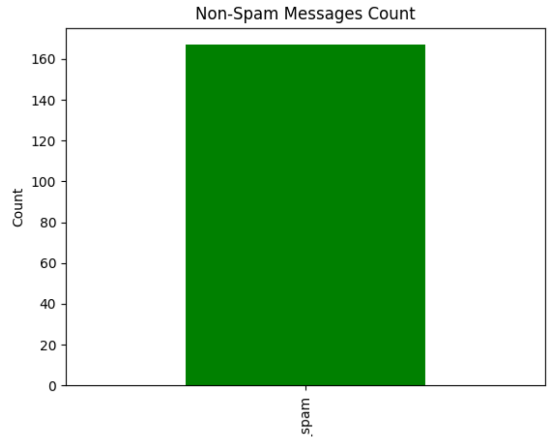
1. **Text Cleaning & Normalization:**
    - Remove punctuation, numbers, special characters, and extra spaces.
    - Normalize Arabic letters (إ, أ, آ → ا).
    - Remove diacritics and elongations.
    - Correct common spelling errors.
2. **Filtering & Deduplication:**
    - Remove incomplete or non-Arabic messages.
    - Remove duplicate messages.
3. **Tokenization & Feature Engineering:**
    - Tokenize text using NLTK.
    - Extract features such as word count, character count, and keyword frequency.

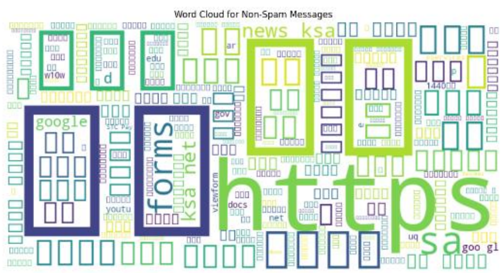
Dataset Visualization (optional but recommended):

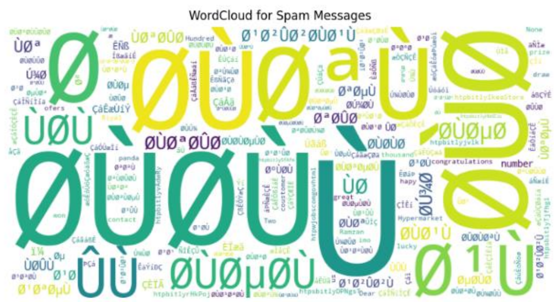
- Pie chart for spam vs ham distribution.
- Histogram of message length.











3. Optimization

1. Hyperparameter Tuning:

- Adjust parameters like learning rate, number of layers, batch size, and number of neurons.
- Use grid search or manual tuning to maximize performance.

2. Data Balancing:

- Handle class imbalance between spam and ham messages.
  - Oversampling: Duplicate minority class (spam).
  - Undersampling: Reduce majority class (ham).

3. Preprocessing Optimization:

- Removing noise, normalizing text, and feature engineering improved model's ability to detect patterns.

4. Evaluation Metrics:

- F1-Score: Balances precision and recall (important to detect spam without missing legitimate messages).
- AUC (Area Under Curve): Measures overall classification performance.

---

4. Testing & Results

Test Dataset Metrics:

Metric	**Value**
Accuracy	0.92
Precision	0.91
Recall	0.90
F1-Score	0.905

Interpretation:

- High accuracy indicates effective distinction between spam and non-spam messages.
- Balanced F1-score shows the model detects spam reliably while minimizing false positives.

---

5. API Deployment (Optional)

Implementation:

- Converted the model into an API using **FastAPI**.
- Endpoint: /predict/
- Example input/output:

```
{
  "text": "تحقق من رصيدك الآن!",
  "classification": "spam",
  "confidence": 92.5
}
```